PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-083310

(43)Date of publication of application : 31.03.1998

(51)Int.Cl. G06F 9/445
G06F 9/06
G06F 12/14

(21)Application number : 09-151768

(22)Date of filing : 10.06.1997

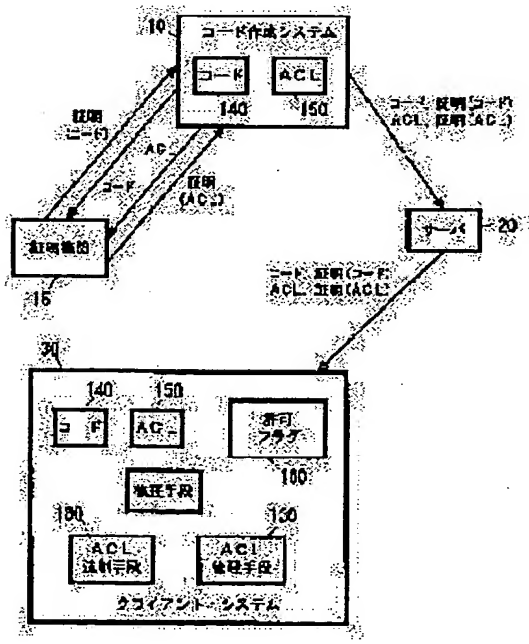
(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>

(72)Inventor : DAN ASIT
RAMASWAMI RAJIV
SITARAM DINKAR

(30)Priority

Priority number : 96 661517 Priority date : 11.06.1996 Priority country : US

(54) PROGRAM CODE DISTRIBUTING METHOD AND ITS SYSTEM



(57)Abstract:

PROBLEM TO BE SOLVED: To provide an authentification system for allowing a relied third party to confirm the author of a program and to sign a certification for guaranteeing the perfection of the program.

SOLUTION: A program code 140 is capsuled together with a guarantee and access control list(ACL) 150. ACL 150 describes an allowable condition and a resource required by the code 140. A forcing mechanism assigns the allowanble condition of a system and a resource according to ACL 150. For example, a code preparing system 10 communicates with a certifying organization 15 being the relied third party. The organization 15 issues the cerificate of the code 140 and the certificate of ACL 150 of the code 140. Once the certificate is issued, nobody can change the code 140 and ACL 150 without invalidating the certificate. The code 140, its ACL 150 and their certificates are stored in a server.

CLAIMS

[Claim(s)]

[Claim 1] The distribution approach of the program code which is the distribution approach of a program code and is characterized by to describe a resource and permissive conditions required for the harmless actuation in which the above-mentioned program code was inspected possible [reading] by the computer in certification of the above-mentioned third person by whom reliance is done including the step which offers the certification of the third person trusted to a reception system.

[Claim 2] The distribution approach of the program code which contains the step which encapsulates the above-mentioned certification with a program code in claim 1.

[Claim 3] The distribution approach of the program code which contains the resource of the above-mentioned reception system, and the step which assigns permissive conditions in claim 1 so that the step in which the above-mentioned reception system reads the above-mentioned certification, and the permissive conditions by which it was specified during the above-mentioned certification may not be exceeded.

[Claim 4] The distribution approach of the program code containing the step which denies or affirms access and permissive conditions of the above-mentioned program code to the resource of the above-mentioned reception system in claim 2 according to the option which the user chose in relation to the above-mentioned certification.

[Claim 5] The distribution approach of the program code characterized by offering the above-mentioned resource described possible [reading] by computer, and permissive conditions in a code format in claim 1 in the above-mentioned reception system.

[Claim 6] The distribution approach of a program code that the enciphered inspection data are contained in the above-mentioned certification in claim 1, and the above-mentioned reception system is characterized by decoding the above-mentioned inspection data and inspecting the integrity of description of the above-mentioned resource and permissive conditions.

[Claim 7] The distribution approach of the program code characterized by describing both the amount of each resource used by the above-mentioned program code into required description of a resource, and the maximum-permissible specific consumption of the resource in claim 3.

[Claim 8] The distribution approach of the program code characterized by describing the specific function of the above-mentioned reception system accessed by the above-mentioned program code into required description of permissive conditions in claim 3.

[Claim 9] The distribution approach of the program code characterized by describing the functionality of the above-mentioned program code in certification of the above-mentioned third person according to inspection by the above-mentioned third person in claim 1.

[Claim 10] The distribution approach of the program code characterized by the above-mentioned program code being the applet downloaded as a program object from a server in claim 1.

[Claim 11] The distribution approach of the program code characterized by assigning a resource

which is different in claim 1 to the user from whom the above-mentioned program code differed in the above-mentioned reception system, and permissive conditions.

[Claim 12] The distribution approach of a program code that the group of the resource which a user different the account of a top is allowed in claim 11, and permissive conditions is characterized by being determined at the time of install of the above-mentioned program code.

[Claim 13] The distribution approach of a program code that the group of the resource which a user different the account of a top is allowed in claim 11, and permissive conditions is characterized by being determined at the time of activation of the above-mentioned program code.

[Claim 14] In the certification of the program code by the third person who is the distribution approach of a program code and is trusted What described a resource and permissive conditions required for the harmless actuation in which this program code was inspected possible [reading] by computer is put in. The step which encapsulates the certification with the above-mentioned program code, and is offered to a reception system in a code format, The step in which the above-mentioned reception system reads the certification of the above-mentioned code format, The account of a top The step which determines the integrity of the read certification by the above-mentioned reception system, The step which assigns the resource and permissive conditions of the above-mentioned reception system according to the option which the user chose so that the permissive conditions specified in the above-mentioned certification might not be exceeded, after the above-mentioned integrity was inspected, the step which performs the above-mentioned program code according to the above-mentioned assignment -- containing -- the above -- in required description of a resource Both the amounts and maximum-permissible specific consumption of each resource used by the above-mentioned program code are described. In description of the above-mentioned permissive conditions The distribution approach of the program code characterized by describing the specific function of the above-mentioned reception system accessed by the above-mentioned program code.

[Claim 15] The import equipment which imports a program and data to computer system, The operating system which controls actuation of the above-mentioned computer system, The access logic which extracts what described the resource required for the harmless actuation in which the code was inspected possible [reading] by computer from the above-mentioned data, and relates it with a given program, The integrity inspection logic which generates the inspection data in which integrity is shown although it was contained in this access logic and the above-mentioned computer described possible [reading], Connect with the above-mentioned operating system, answer the above-mentioned inspection data, and consumption and specific consumption are pursued and assigned in the above-mentioned reception system about each of many resources. Computer system possessing the compulsive logic it is made not to exceed the assignment specified in the above-mentioned description, and the processor which performs a program code according to the above-mentioned assignment.

[Claim 16] The DS memorized by random access memory is connected to the above-mentioned compulsion logic in claim 15. In this DS The 1st field which pursues the consumed actual resource,

and the 2nd field which pursues the specific consumption of a resource, Computer system characterized by including the 3rd field which memorizes the limit of the resource consumption pulled out from the above-mentioned description, and the 4th field which memorizes the limit of the resource specific consumption pulled out from the above-mentioned description.

[Claim 17] Computer system characterized by including a means to un-encapsulate the above-mentioned description in claim 15 from the package which contains a program code in the above-mentioned access logic, and a means to decode this description.

[Claim 18] Computer system characterized by including a means to deny or affirm code access to the resource of computer system, and permissive conditions in claim 15 according to the option which the user chose in relation to the above-mentioned description into the above-mentioned compulsion logic.

DETAILED DESCRIPTION

[Detailed Description of the Invention]

[0001]

[Field of the Invention] Generally this invention relates to sending software using a distribution system like a network.

[0002]

[Description of the Prior Art] The distribution business of software is the latest big industry. Software is distributed through a diskette and CD-ROM, and came to be gradually distributed through the network today. Furthermore, only through the Internet, from a remote site, a client can download an applet from a server and, recently, it it not only downloads a code, but can be performed. This is a paradigm proposed in Java (Java) and programming language, or other language like Telescript.

[0003] The important problem accompanying the code obtained from others is a problem of security. For example, in a UNIX operating system, a code is performed in the client shell possessing all the privileges of a client. Accessing all the files of a client, sending e-mail, and trying trespass further are also included in such a privilege. Java performs an applet in the environment restricted very much, in order to solve the problem of such an applet. Consequently, the usefulness and functionality of an applet of Java are restricted. On the other hand, the application of Java does not get used to any authentications (authentication) depending on the security offered by the operating system. Therefore, these applications raise the problem of the same security as other codes.

[0004] One format of authentication is proposed by "Trusted Distribution ofSoftware Over the Internet" (A. D. Rubin, Internet Society Symposium - Network and Distributed Security, 1995). There, the integrity of a program is guaranteed, while the third person trusted signs certification and checks the author of a program.

[0005]

[Problem(s) to be Solved by the Invention] Although a client can verify authentication of a code by this approach, this approach does not specify the permissive conditions relevant to a code flexibly, and does not offer how a client forces these permissive conditions automatically, either.

[0006]

[Means for Solving the Problem] The program code is encapsulated, although this invention guarantees the integrity of a program while the third person trusted signs certification and checks the author of a program (that is, related with certification and an access control list). The access control list has described the permissive conditions and the resource which are demanded in code. Furthermore, this invention offers the compulsive mechanism which assigns permissive conditions and a resource to a system according to an access control list.

[0007] A code creation system communicates with the certification engine which is the third person trusted in an example. A certification engine publishes certification of a code, and certification of the access control list about the code. Once certification is published, any persons cannot change the code or access control list, without making the certification into an invalid. A code and its access control list are memorized by the server with those certification. The client which downloads a code or an access control list can inspect the integrity of a code thru/or an access control list, and a system can carry out an access control list so that permissive conditions and a resource may not be exceeded.

[0008]

[Embodiment of the Invention] Drawing 1 is the block diagram of the code sending according to this invention, and a certification system. One or more code creation system 10, certification engine 15, one or more servers 20, and one or more client systems 30 are shown there. A client system is connected to a server through a usual wide area network (WAN) or a usual local area network (LAN). A code creation system contains the program code 140 to prove with a certification engine, and the access control list (ACL) 150 about the code. A certification engine offers the public key K which is used widely and known by the client system. Furthermore, the certification engine holds the private key P which only it knows. In order to offer the certification of a code, a certification engine creates certification including the code hash (hash) of a code name and a code, and signs it using a private key. Now, the certification cannot be changed, without making a signature of a certification engine into an invalid. Similarly, a certification engine also signs the certification for access control lists relevant to a code (supposing it wishes, you may make it sign only one certification for the both sides of a code and its access control list). In the example of drawing 1 , a client system receives certification, an access control list, and a code through WAN or LAN. However, please change to the approach and notice a client system about the point that the proved code is receivable, through a dismountable storage. Such a storage is read by local import equipment like a floppy drive or an optical disk drive.

[0009] The client system shown in drawing 2 includes the inspection means 110, the access control list (ACL) management tool 130, the activation means 170, the access control list (ACL) compulsion means 180, and the client interface 190. Furthermore, a client system contains the usual CPU and

the related control logic 162, the communication link controller / network interface 194, the CD-ROM / floppy disk subsystem 196, and other resources (for example, a network access module, a display subsystem, and various kinds of special purpose adapters). Although it will probably be clear for this contractor that a client system contains many other usual components, they are not explained to a detail here.

[0010] As for the access control list management tool 130 and the access control list compulsion means 180, it is desirable to carry out as a program code. The actuation is explained to a detail later. An activation means is the usual part of the operating system (not shown) of a client, and it is used in order to perform the code imported on the client system. The client interface 190 can be carried out as a front of a screen graphical user interface, and enables communication with the access control list management tool 130 and a client. (For example, by it, a client can order the access control list management tool 130 whether to allow program access to the specified resource, or not to allow, or can manage program access.) As for the inspection means 110, it is desirable to carry out as a program code. It inspects authentication of the code imported with the access control list including a decode module. Furthermore, an inspection means contains a hashing (hashing) module. This module checks the integrity of the code and access control list which were imported.

[0011] "A code / access control list, and certification" If 100 downloads, the sign of the certification engine which the inspection means 110 has after certification will inspect first whether it is the right (a certification engine's known public key is used). Next, it inspects whether an inspection means calculates the code hash of a code/access control list, and its it corresponds with the value under certification. When a sign is not right, the code and access control list are refused (refusal step 120). (namely, when a hash is not in agreement) If inspection is O.K., the access control list management tool 130 will be called. An access control list management tool displays an access control list (it explains later) on a client through the client interface 190, and a client confirms [to which each item in an access control list is permitted / or or] whether authorization is carried out. The access control list management tool 130 memorizes a code 140, and memorizes an access control list 150 with the authorization flag 160 as it was ordered from the client through the client interface.

[0012] There are a file system, a specific file and a logic resource like a system call, and a physical resource like assignment of desk space, disk access, and main memory and access to various kinds of system controllers and adapters as resource which can control access by the access control list compulsion means 180. About access to a logic resource (function of a client system), the authorization flag 160 indicates [to which each item is permitted / or or] whether authorization is carried out. About access to a physical resource, in order to show the maximum permissible dose or maximum-permissible specific consumption, an authorization flag can be used.

[0013] Furthermore, as for drawing 2 , the part from which the client system differed shows how it operates mutually. In a multiple user client system, each user can have each authorization flag group. Furthermore, an environment variable can be put in into an access control list. By changing an environment variable during activation, each user can customize an access privilege. An access

control list and an authorization flag are memorized to a security field, and read-out and updating of this field are a privilege which the access control list compulsion means 180 is allowed.

[0014] An access control list management tool may be called by the client at the time of arbitration, in order to display or change the authorization conditions of an access control list. A code is performed by the activation means 170. Before allowing access to a resource, an activation means calls an access control list compulsion means, and checks the effectiveness of access. This is attained by the approach the activation means 170 inserts into a code the trap to the test routine which calls an access control list compulsion means. Actuation of an access control list compulsion means is explained to a detail later.

[0015] In a system, if needed, separately, it can combine and a code and its access control list can be downloaded. For example, as a plan of a code creation system, although an access control list is offered to all clients for nothing, the code itself can be made into the charge.

[0016] Drawing 3 shows an access control list. An access control list consists of two parts. That is, they are the physical resource table 200 containing the physical resource demanded in code, and the logic resource table 250 containing the authorization conditions demanded in code, and a logic resource.

[0017] The line about each resource is in the physical resource table 200, and the physical resource name 205, a resource attribute 210, the maximum-permissible specific consumption 215, and the maximum permissible dose 220 are in this line. A resource attribute 210 is used when two or more attributes are in a physical unit or a resource disk. For example, it is a case as there are a tooth space and the number of I/O about storage. The maximum-permissible specific consumption 215 and the maximum permissible dose 220 are a resource, the maximum-permissible specific consumption of an attribute, and maximum-permissible consumption.

[0018] The logic resource table 250 contains one line about each call to the external routine (called a logic resource) demanded in code. There are the logic resource name 255 and a parameter list 260 in each line. A parameter list 260 points to the list of parameter entries 265. Each parameter entry 265 specifies the set of the effective parameter range. That is, the set is a set of the effective parameter value as a combination. In the parameter entry 265, the nextPE field 280 indicating the following parameter entry is also included. The parameter range of each parameter includes the two fields. That is, it is the parameter value 275 which specifies the scope of the parameter as the parameter type 270. About a string parameter, the parameter type 270 is STR, and parameter value 275 is a list of regular expressions which specify the effective format about the string. About an integer parameter, the parameter type 270 is INT, and parameter value 275 is a list of integer range. The identifier of an environment variable can be put in into parameter value 275, and permuting the value of the environment variable in that case at the time of activation is assumed.

[0019] The authorization conditions and resource which were specified about the code in the access control list are offered, and the access control list compulsion means in a client ensures that the addition of an authorization condition / resource is not allowed.

[0020] Compulsion of an access control list may be static, or may be dynamic. In static compulsion,

before a code is performed, compulsion is performed extensively, and compulsion while the code is performed is not needed. In dynamic compulsion, while the code is performed, coercion must be exerted. The certification engine itself inspects an access control list, and if it is guaranteed that the excess in a code does not arise, the compulsive function of an access control list will become unnecessary by the client system.

[0021] Drawing 4 shows the DS used by the access control list compulsion means 180. One line is shown in the execution-time physics resource table 300 about each resource. A resource name 300, a resource attribute 310, the maximum-permissible specific consumption 315, and the maximum permissible dose 320 are the copies of the field where the physical resource table 200 corresponds. It is actually used in order to pursue the actual specific consumption of specific consumption 325 and the code [actually / field / of the amount used 330 / each] at the time of activation respectively, and actual consumption. The line about each logic resource needed is shown in the execution-time logic resource table 350. An execution-time logic resource table is the copy of a logic resource table, and the authorization flag is added to it. An authorization flag shows [to which the combination is permitted by the access control list management tool / or or] whether authorization is carried out about each effective combination of a parameter. The logic resource name 355 is the copy of the field where the logic resource table 250 corresponds, and a parameter list 360 points to the list of execution-time parameter entries 365. The parameter type 370 and parameter value 375 of an execution-time parameter entry are the copy of the field where it corresponds in a logic resource table. Yes which shows [to which this execution-time parameter entry 365 is permitted / or or] whether authorization is carried out, the nextPE field 380 points to the following execution-time parameter entry 365, and the authorization flag 385 is set as the no. Furthermore, an access control list compulsion means pursues 395 at the code initiation time.

[0022] Drawing 5 shows how the authorization conditions of a logic resource are forced by the access control list compulsion means. This pass is always called, when a code performs the call to an external function. At step 410, the access control list compulsion means 180 determines the location of the number of parameters, the value of a parameter, and the identifier of the function currently called. The actual approach changes with processors. For example, in Java, these are placed on the operand stack. At step 415, an access control list compulsion means determines the location of the line of the function currently called in the execution-time logic access table 350, and determines the location of the first execution-time parameter entry 365 where the authorization flag is set as yes. When a function name is not found or such an execution-time parameter entry 365 does not exist, an access control list compulsion means progresses to step 470, and it is shown that a call is not allowed and it ends. At step 420, an access control list compulsion means determines the location of the value of the first parameter, and makes it a current parameter. At step 425, an access control list compulsion means confirms whether the value of a current parameter is allowed by the execution-time parameter entry 365. If the value is allowed, it will be confirmed at step 430 whether an access control list compulsion means has a parameter more. If there is furthermore a parameter, at step 435, an access control list compulsion means will set a current parameter as the

following parameter, and will return to step 425. At step 430, if there is no parameter beyond it, an access control list compulsion means progresses to step 450, will display the purport allowed a call and will be completed.

[0023] If the test of admissibility fails in step 425, an access control list compulsion means will progress to step 445 at it. There, an access control list compulsion means checks the execution-time logic resource table 350, and other execution-time parameter entries by which the authorization flag 385 is set to yes are looked for. If there is such no execution-time parameter entry, an access control list compulsion means progresses to step 470, will display the purport which is not allowed a call and will be completed. If there is such an execution-time parameter entry 365, an access control list compulsion means will progress to step 420.

[0024] Drawing 6 shows how physical requirements are forced by the access control list compulsion means. Before an access control list compulsion means assigns a resource, it is called at step 500. Two parameters, i.e., the demanded amount (REQAMT) of resources and consumption anticipation time amount (COMPT), are offered. About disk I/O, REQAMT is the number of disk I/O, and COMPT is anticipation time amount which disk I/O completes. It is confirmed whether whether an access control list compulsion means' determining the line in the execution-time physics resource table 300 about this resource, and the maximum permissible dose's 320 being specified at step 505, and the thing which actually added REQAMT to the amount used 330 exceed the maximum permissible dose 320. If it exceeds, it will display the purport which is not allowed consumption at step 527. When not exceeding the maximum permissible dose 320, an access control list compulsion means calculates the estimated consumption rate of this resource at step 510 according to / (actually amount-used 330 + REQAMT) (this time - code initiation time 395 + COMPT). At step 515, it is confirmed whether an access control list compulsion means has whether the maximum-permissible specific consumption 315 is specified and the planned specific consumption larger than the maximum-permissible specific consumption 315. If the maximum-permissible specific consumption 315 is not specified or an estimated consumption rate is not larger than it, an access control list compulsion means displays the purport allowed consumption, and is completed. It is the /(actually amount-used 300 + REQAMT) maximum-permissible specific consumption 315 about delay required to perform [for an access control list compulsion means to be step 530 if an estimated consumption rate is large, and] this actuation. - (this time -code initiation time 395 - COMPT) While it follows and calculates and only the calculated time delay displays the purport for which consumption must be delayed, a required time delay is returned.

[0025] After a resource is consumed, an access control list compulsion means is called at step 550 using the parameter which specifies resource consumption (CONSAMT). The called access control list compulsion means actually updates the amount used 330 with an activity ratio 325.

[0026] Furthermore, a different resource and permissive conditions can be assigned to a different user by the reception system about the same code. It can be performed by combining a privilege with the resource and permissive conditions which a different user was allowed by the access control list compulsion means at the code paying attention to the granted privilege, when a code is

installed. In this case, probably, the set of a resource and permissive conditions needs to memorize separately for every user. When the code is performed, a resource and permissive conditions will be forced with the user base. On the other hand, when determining a resource and permissive conditions during activation of a code, it can determine by combining a privilege with the resource and permissive conditions which a different user was allowed by the access control list compulsion means at the code paying attention to the granted privilege.

[0027] Drawing 7 is a pseudocode which shows the initialization action of integrity inspection conducted by the client system, activation, and compulsion. Note that instantiation of the DS of various kinds of tables explained until now, a list, and a flag and others is carried out in the memory (for example, volatile random access memory, a disk, or these should put together) of a client system. As mentioned above, as for an access control list compulsion means and an access control list management tool, it is desirable to carry out as a program code. It is incorporated whether these program codes are linked to the operating system of a client system. Drawing 8 shows the pseudocode of an access control list management tool. Drawing 9 shows the pseudocode of an access control list compulsion means.

[0028] Although this invention has been explained based on an example, it is thought that various kinds of modification and improvements by this contractor are possible. Therefore, the example of this invention is a mere example and he should understand what must not be interpreted restrictively. It cannot be overemphasized that the range of this invention is limited by the generic claim.

[0029] As a conclusion, the following matters are indicated about the configuration of this invention.

(1) The distribution approach of the program code which is the distribution approach of a program code and is characterized by to describe a resource and permissive conditions required for the harmless actuation in which the above-mentioned program code was inspected possible [reading] by computer in certification of the above-mentioned third person by whom reliance is done including the step which offers the certification of the third person trusted to a reception system.

(2) The distribution approach of the program code which contains the step which encapsulates the above-mentioned certification with a program code in the above (1).

(3) The distribution approach of the program code which contains the resource of the above-mentioned reception system, and the step which assigns permissive conditions in the above (1) so that the step in which the above-mentioned reception system reads the above-mentioned certification, and the permissive conditions by which it was specified during the above-mentioned certification may not be exceeded.

(4) The distribution approach of the program code containing the step which denies or affirms access and permissive conditions of the above-mentioned program code to the resource of the above-mentioned reception system in the above (2) according to the option which the user chose in relation to the above-mentioned certification.

(5) The distribution approach of the program code characterized by offering the above-mentioned resource described possible [reading] by computer, and permissive conditions in a code format in

the above (1) in the above-mentioned reception system.

(6) The distribution approach of a program code that the enciphered inspection data are contained in the above-mentioned certification in the above (1), and the above-mentioned reception system is characterized by decoding the above-mentioned inspection data and inspecting the integrity of description of the above-mentioned resource and permissive conditions.

(7) The distribution approach of the program code characterized by describing both the amount of each resource used by the above-mentioned program code into required description of a resource, and the maximum-permissible specific consumption of the resource in the above (3).

(8) The distribution approach of the program code characterized by describing the specific function of the above-mentioned reception system accessed by the above-mentioned program code into required description of permissive conditions in the above (3).

(9) The distribution approach of the program code characterized by describing the functionality of the above-mentioned program code in certification of the above-mentioned third person according to inspection by the above-mentioned third person in the above (1).

(10) The distribution approach of the program code characterized by the above-mentioned program code being the applet downloaded as a program object from a server in the above (1).

(11) The distribution approach of the program code characterized by assigning a resource which is different in the above (1) to the user from whom the above-mentioned program code differed in the above-mentioned reception system, and permissive conditions.

(12) The distribution approach of a program code that the group of the resource which a user different the account of a top is allowed in the above (11), and permissive conditions is characterized by being determined at the time of install of the above-mentioned program code.

(13) The distribution approach of a program code that the group of the resource which a user different the account of a top is allowed in the above (11), and permissive conditions is characterized by being determined at the time of activation of the above-mentioned program code.

(14) In the certification of the program code by the third person who is the distribution approach of a program code and is trusted What described a resource and permissive conditions required for the harmless actuation in which this program code was inspected possible [reading] by computer is put in. The step which encapsulates the certification with the above-mentioned program code, and is offered to a reception system in a code format, The step in which the above-mentioned reception system reads the certification of the above-mentioned code format, The account of a top The step which determines the integrity of the read certification by the above-mentioned reception system, The step which assigns the resource and permissive conditions of the above-mentioned reception system according to the option which the user chose so that the permissive conditions specified in the above-mentioned certification might not be exceeded, after the above-mentioned integrity was inspected, the step which performs the above-mentioned program code according to the above-mentioned assignment -- containing -- the above -- in required description of a resource Both the amounts and maximum-permissible specific consumption of each resource used by the above-mentioned program code are described. In description of the above-mentioned permissive

conditions The distribution approach of the program code characterized by describing the specific function of the above-mentioned reception system accessed by the above-mentioned program code.
(15) The import equipment which imports a program and data to computer system, The operating system which controls actuation of the above-mentioned computer system, The access logic which extracts what described the resource required for the harmless actuation in which the code was inspected possible [reading] by computer from the above-mentioned data, and relates it with a given program, The integrity inspection logic which generates the inspection data in which integrity is shown although it was contained in this access logic and the above-mentioned computer described possible [reading], Connect with the above-mentioned operating system, answer the above-mentioned inspection data, and consumption and specific consumption are pursued and assigned in the above-mentioned reception system about each of many resources. Computer system possessing the compulsive logic it is made not to exceed the assignment specified in the above-mentioned description, and the processor which performs a program code according to the above-mentioned assignment.
The DS memorized by random access memory is connected to the above-mentioned compulsion logic in the above (15). (16) In this DS The 1st field which pursues the consumed actual resource, and the 2nd field which pursues the specific consumption of a resource, Computer system characterized by including the 3rd field which memorizes the limit of the resource consumption pulled out from the above-mentioned description, and the 4th field which memorizes the limit of the resource specific consumption pulled out from the above-mentioned description.
(17) Computer system characterized by including a means to un-encapsulate the above-mentioned description in the above (15) from the package which contains a program code in the above-mentioned access logic, and a means to decode this description.
(18) Computer system characterized by including a means to deny or affirm code access to the resource of computer system, and permissive conditions in the above (15) according to the option which the user chose in relation to the above-mentioned description into the above-mentioned compulsion logic.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]
[Drawing 1] It is the block diagram showing code distribution / certification system according to this invention.
[Drawing 2] In order that each part of a client system may perform the function explained in the example, it is drawing showing how it operates mutually.
[Drawing 3] It is drawing showing an access control list (ACL).
[Drawing 4] It is drawing showing the DS used by the access control list (ACL) compulsion means.
[Drawing 5] It is drawing showing how the permissive conditions of a logic resource are forced by the access control list (ACL) compulsion means.

[Drawing 6] It is drawing showing how the limit of a physical resource is forced by the access control list (ACL) compulsion means.

[Drawing 7] It is drawing having shown the initialization action of integrity inspection of a client system, activation, and compulsion by the pseudocode.

[Drawing 8] It is drawing showing the pseudocode of an access control list (ACL) management tool.

[Drawing 9] It is drawing showing the pseudocode of an access control list (ACL) compulsion means.

[Description of Notations]

10 Code Creation System

15 Certification Engine

20 Server

30 Client System

100 Code / Access Control List, and Certification

110 Inspection Means

120 Refusal Step

130 Access Control List Management Tool

140 Code

150 Access Control List

160 Authorization Flag

162 CPU and Related Control Logic

170 Activation Means

180 Access Control List Compulsion Means

190 Client Interface

194 Communication Link Controller / Network Interface

196 CD-ROM / Floppy Disk Subsystem

200 Physical Resource Table

205 Physical Resource Name

210 Resource Attribute

215 Maximum-Permissible Specific Consumption

220 Maximum Permissible Dose

250 Logic Resource Table

255 Logic Resource Name

260 Parameter List

265 Parameter Entry

270 Parameter Type

275 Parameter Value

280 NextPE (Following Parameter Entry) Field

300 Execution-Time Physics Resource Table

305 Physical Resource Name

310 Resource Attribute

315 Maximum-Permissible Specific Consumption

320 Maximum Permissible Dose

325 It is Actually Specific Consumption.

330 It is Actually the Amount Used.

350 Execution-Time Logic Resource Table

355 Logic Resource Name

360 Parameter List

365 Execution-Time Parameter Entry

370 Parameter Type

375 Parameter Value

380 NextPE (Following Parameter Entry) Field

385 Authorization Flag

395 Code Initiation Time

---

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－83310

(43)公開日　平成10年(1998) 3月31日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	9/445			Ｇ０６Ｆ	9/06	４２０Ｊ
	9/06	５５０				５５０Ｚ
	12/14	３１０			12/14	３１０Ｋ
						３１０Ｚ

審査請求　未請求　請求項の数18　ＯＬ　（全 16 頁）

(21)出願番号　特願平9－151768

(22)出願日　平成９年(1997) ６月10日

(31)優先権主張番号　０８／６６１５１７

(32)優先日　1996年６月11日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　アスィット・ダン
アメリカ合衆国10604、　ニューヨーク州　ウェスト　ハリソン　ゲインスボーグ　アヴェニュー　75

(74)代理人　弁理士　坂口　博　（外１名）

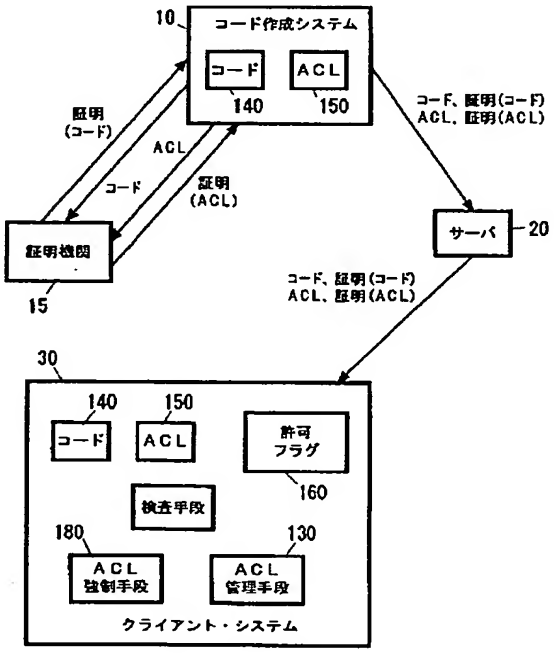
最終頁に続く

(54)【発明の名称】　プログラム・コードの配布方法及びシステム

(57)【要約】

【課題】　信頼される第三者がプログラムの作者を確認しプログラムの完全性を保証するための証明に署名する認証方式を提供すること。

【解決手段】　プログラム・コードは証明およびアクセス制御リスト（ＡＣＬ）と共にカプセル化されている。アクセス制御リストはコードによって要求される許容条件とリソースを記述する。強制メカニズムはアクセス制御リストに従ってシステムの許容条件とリソースを割り振る。実施例において、コード作成システムは、信頼される第三者である証明機関と通信する。証明機関はコードの証明と、そのコードのアクセス制御リストの証明とを発行する。一度、証明が発行されると、いかなる者もその証明を無効にすることなくコードまたはアクセス制御リストを変更することはできない。コード、そのアクセス制御リスト、およびそれらの証明はサーバ中に記憶される。



1

【特許請求の範囲】

【請求項１】プログラム・コードの配布方法であって、信頼される第三者の証明を受け取りシステムへ提供するステップを含み、上記プログラム・コードの検査された無害の動作に必要なリソースと許容条件を、上記信頼される第三者の証明の中にコンピュータで読み取り可能に記述したことを特徴とする、プログラム・コードの配布方法。

【請求項２】請求項１において、上記証明をプログラム・コードと共にカプセル化するステップを含む、プログラム・コードの配布方法。

【請求項３】請求項１において、上記受け取りシステムが上記証明を読み取るステップと、上記証明中に指定された許容条件を超過しないように上記受け取りシステムのリソースと許容条件を割り振るステップとを含む、プログラム・コードの配布方法。

【請求項４】請求項２において、上記証明に関連してユーザが選択したオプションに従って、上記受け取りシステムのリソースに対する上記プログラム・コードのアクセスと許容条件を否定または肯定するステップを含む、プログラム・コードの配布方法。

【請求項５】請求項１において、コンピュータで読み取り可能に記述した上記リソースと許容条件が暗号形式で上記受け取りシステムへ提供されることを特徴とする、プログラム・コードの配布方法。

【請求項６】請求項１において、暗号化された検査データが上記証明の中に含まれており、上記受け取りシステムが上記検査データを解読して上記リソースと許容条件の記述の完全性を検査することを特徴とする、プログラム・コードの配布方法。

【請求項７】請求項３において、必要なリソースの記述の中に、上記プログラム・コードによって使用される各リソースの量と、そのリソースの最大許容消費率の両方が記述されていることを特徴とする、プログラム・コードの配布方法。

【請求項８】請求項３において、必要な許容条件の記述の中に、上記プログラム・コードによってアクセスされる上記受け取りシステムの特定の機能が記述されていることを特徴とする、プログラム・コードの配布方法。

【請求項９】請求項１において、上記プログラム・コードの機能性が、上記第三者による検査に従って上記第三者の証明の中に記述されていることを特徴とする、プログラム・コードの配布方法。

【請求項１０】請求項１において、上記プログラム・コードが、サーバからプログラム・オブジェクトとしてダウンロードされるアプレットであることを特徴とする、プログラム・コードの配布方法。

【請求項１１】請求項１において、上記受け取りシステムの中で上記プログラム・コードの異なったユーザに対しては異なったリソースと許容条件を割り振ることを特

2

徴とする、プログラム・コードの配布方法。

【請求項１２】請求項１１において、上記異なったユーザに許されるリソースと許容条件の組が、上記プログラム・コードのインストール時に決定されることを特徴とする、プログラム・コードの配布方法。

【請求項１３】請求項１１において、上記異なったユーザに許されるリソースと許容条件の組が、上記プログラム・コードの実行時に決定されることを特徴とする、プログラム・コードの配布方法。

【請求項１４】プログラム・コードの配布方法であって、信頼される第三者によるプログラム・コードの証明の中に、該プログラム・コードの検査された無害の動作に必要なリソースと許容条件をコンピュータで読み取り可能に記述したものを入れ、その証明を上記プログラム・コードと共にカプセル化して暗号形式で受け取りシステムへ提供するステップと、上記受け取りシステムが上記暗号形式の証明を読み取るステップと、上記読み取られた証明の完全性を上記受け取りシステムによって決定するステップと、上記完全性が検査された後に、上記証明の中で指定された許容条件を超過しないようにユーザが選択したオプションに従って上記受け取りシステムのリソースと許容条件を割り振るステップと、上記割り振りに従って上記プログラム・コードを実行するステップとを含み、上記必要なリソースの記述の中には、上記プログラム・コードによって使用される各リソースの量とその最大許容消費率との両方が記述され、上記許容条件の記述の中には、上記プログラム・コードによってアクセスされる上記受け取りシステムの特定の機能が記述されていることを特徴とする、プログラム・コードの配布方法。

【請求項１５】コンピュータ・システムへプログラムとデータを移入する移入装置と、上記コンピュータ・システムの動作を制御するオペレーティング・システムと、コードの検査された無害の動作に必要なリソースをコンピュータで読み取り可能に記述したものを、上記データから抽出してそれを所与のプログラムに関連づけるアクセス・ロジックと、該アクセス・ロジックに含まれて、上記コンピュータで読み取り可能に記述したものの完全性を示す検査データを生成する完全性検査ロジックと、上記オペレーティング・システムに接続され、上記検査データに応答して多数のリソースの各々について上記受け取りシステムの中で消費量と消費率を追跡し割り振って、上記記述の中に指定された割り振りを超過しないようにする強制ロジックと、上記割り振りに従ってプログラム・コードを実行するプロセッサと、を具備するコンピュータ・システム。

【請求項１６】請求項１５において、ランダム・アクセス・メモリに記憶されたデータ構造が上記強制ロジックに接続されており、該データ構造の中には、消費された実際のリソースを追跡する第１のフィールドと、リソー

スの消費率を追跡する第２のフィールドと、上記記述から引き出されたリソース消費の限度を記憶する第３のフィールドと、上記記述から引き出されたリソース消費率の限度を記憶する第４のフィールドとが含まれていることを特徴とする、コンピュータ・システム。

【請求項１７】請求項１５において、上記アクセス・ロジックの中に、プログラム・コードを含むパッケージから上記記述を非カプセル化する手段と該記述を解読する手段とが含まれていることを特徴とする、コンピュータ・システム。

【請求項１８】請求項１５において、上記強制ロジックの中に、上記記述に関連してユーザが選択したオプションに従ってコンピュータ・システムのリソースに対するコード・アクセスと許容条件を否定または肯定する手段が含まれていることを特徴とする、コンピュータ・システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、一般的には、ネットワークのような頒布システムを使用してソフトウェアを送付することに関する。

【０００２】

【従来の技術】ソフトウェアの頒布業は最近の大きな産業である。今日、ソフトウェアはディスケットおよびＣＤ－ＲＯＭを介して頒布され、また次第にネットワークを介して頒布されるようになった。さらに、最近では、単にインターネットを介して遠隔サイトからコードをダウンロードするだけでなく、クライアントはサーバからアプレットをダウンロードして実行することができる。これは、ジャバ（Ｊａｖａ）・プログラミング言語、またはテレスクリプトのような他の言語で提案されているパラダイムである。

【０００３】他から得られたコードに伴う重要な問題は、セキュリティの問題である。たとえば、ＵＮＩＸオペレーティング・システムでは、コードはクライアントのすべての特権を具備しているクライアント・シェルの中で実行される。そのような特権には、クライアントのすべてのファイルにアクセスすること、メールを送ること、さらには不法侵入を試みることも含まれる。ジャバは、このようなアプレットの問題を解決するために、非常に制限された環境でアプレットを実行する。その結果、ジャバのアプレットの有用性と機能性は制限される。他方、ジャバのアプリケーションはオペレーティング・システムによって提供されるセキュリティに依存し、どのような認証（ａｕｔｈｅｎｔｉｃａｔｉｏｎ）にもなじまない。したがって、それらアプリケーションは他のコードと同じようなセキュリティの問題を提起する。

【０００４】認証の１つの形式は、「Ｔｒｕｓｔｅｄ Ｄｉｓｔｒｉｂｕｔｉｏｎ ｏｆＳｏｆｔｗａｒｅ Ｏｖｅｒ ｔｈｅ Ｉｎｔｅｒｎｅｔ」（Ａ．Ｄ．Ｒｕｂｉｎ，Ｉｎｔｅｒｎｅｔ Ｓｏｃｉｅｔｙ Ｓｙｍｐｏｓｉｕｍ ｏｎ Ｎｅｔｗｏｒｋ ａｎｄ Ｄｉｓｔｒｉｂｕｔｅｄ Ｓｅｃｕｒｉｔｙ，１９９５）に提案されている。そこでは、信頼される第三者が証明に署名して、プログラムの作者を確認するとともにプログラムの完全性を保証している。

【０００５】

【発明が解決しようとする課題】この方法によって、クライアントはコードの認証を検証できるが、この方法は、コードに関連した許容条件を柔軟に指定するものではなく、またクライアントがこれらの許容条件を自動的に強制する方法を提供するものでもない。

【０００６】

【課題を解決するための手段】本発明は、信頼される第三者が証明に署名して、プログラムの作者を確認するとともにプログラムの完全性を保証するものであるが、プログラム・コードはカプセル化されている（すなわち、証明およびアクセス制御リストと関連づけられている）。アクセス制御リストはコードによって要求される許容条件とリソースを記述している。さらに、本発明は、アクセス制御リストに従ってシステムに許容条件とリソースを割り振る強制メカニズムを提供する。

【０００７】実施例において、コード作成システムは、信頼される第三者である証明機関と通信する。証明機関はコードの証明と、そのコードに関するアクセス制御リストの証明を発行する。一度、証明が発行されると、いかなる者も、その証明を無効にすることなしに、そのコードまたはアクセス制御リストを変更することはできない。コードおよびそのアクセス制御リストは、それらの証明と共にサーバに記憶される。コードまたはアクセス制御リストをダウンロードするクライアントは、コードないしアクセス制御リストの完全性を検査することができ、システムは許容条件とリソースを超過しないようにアクセス制御リストを実施することができる。

【０００８】

【発明の実施の形態】図１は、本発明に従うコード送付および証明システムのブロック図である。そこには１つまたは複数のコード作成システム１０、証明機関１５、１つまたは複数のサーバ２０、および１つまたは複数のクライアント・システム３０が示される。クライアント・システムは通常の広域ネットワーク（ＷＡＮ）またはローカル・エリア・ネットワーク（ＬＡＮ）を介してサーバへ接続される。コード作成システムは、証明機関によって証明してもらいたいプログラム・コード１４０と、そのコードに関するアクセス制御リスト（ＡＣＬ）１５０とを含む。証明機関は、クライアント・システムによって広く使用され知られている公用キーＫを提供する。さらに、証明機関は、それだけが知っている私用キーＰを保持している。コードの証明を提供するために

は、証明機関はコード名とコードの暗号ハッシュ（ｈａｓｈ）を含む証明を作成し、私用キーを使用してそれに署名する。今や、その証明は証明機関の署名を無効にすることなしに変更することはできない。同様に、証明機関は、コードに関連したアクセス制御リスト用の証明にも署名する（もし望むなら、コードとそのアクセス制御リストの双方のために、ただ１つの証明にサインするようにしてもよい）。図１の実施例では、クライアント・システムは、ＷＡＮまたはＬＡＮを介して証明、アクセス制御リスト、およびコードを受け取る。しかし、その方法に替えて、クライアント・システムは、取り外し可能記憶媒体を介して、証明されたコードを受け取ることができる点に注意されたい。このような記憶媒体は、フロッピ・ドライブまたは光学ディスク・ドライブのようなローカル移入装置によって読み取られる。

【０００９】図２に示されたクライアント・システムは検査手段１１０、アクセス制御リスト（ＡＣＬ）管理手段１３０、実行手段１７０、アクセス制御リスト（ＡＣＬ）強制手段１８０、およびクライアント・インタフェース１９０を含む。さらに、クライアント・システムは通常のＣＰＵおよび関連制御ロジック１６２、通信コントローラ／ネットワーク・インタフェース１９４、ＣＤ－ＲＯＭ／フロッピ・ディスク・サブシステム１９６、および他のリソース（たとえば、ネットワーク・アクセス・モジュール、ディスプレイ・サブシステム、および各種の特殊目的アダプタ）を含む。当業者にとって、クライアント・システムが他の多くの通常のコンポーネントを含むことは明らかであろうが、それらについては、ここで詳細に説明しない。

【００１０】アクセス制御リスト管理手段１３０とアクセス制御リスト強制手段１８０は、プログラム・コードとして実施するのが望ましい。その動作については後で詳細に説明する。実行手段はクライアントのオペレーティング・システム（図示されず）の通常の部分であり、クライアント・システム上で、移入されたコードを実行するために使用される。クライアント・インタフェース１９０は画面グラフィカル・ユーザ・インタフェースのフロントとして実施することができ、アクセス制御リスト管理手段１３０とクライアントとの連絡を可能にする。（たとえば、それによって、クライアントは、指定されたリソースへのプログラム・アクセスを許すか許さないかをアクセス制御リスト管理手段１３０に命令したり、プログラム・アクセスを管理したりすることができる。）検査手段１１０はプログラム・コードとして実施することが望ましい。それは解読モジュールを含み、アクセス制御リストと共に移入されたコードの認証を検査する。さらに、検査手段はハッシング（ｈａｓｈｉｎｇ）モジュールを含む。このモジュールは、移入されたコードとアクセス制御リストの完全性をチェックする。

【００１１】「コード／アクセス制御リストおよび証明」１００がダウンロードされると、まず検査手段１１０は証明の上にある証明機関のサインが正しいかどうかを検査する（証明機関の既知の公用キーを使用する）。次に、検査手段はコード／アクセス制御リストの暗号ハッシュを計算して、それが証明中の値と一致するかどうかを検査する。サインが正しくないとき（すなわち、ハッシュが一致しないとき）、そのコードとアクセス制御リストは拒絶される（拒絶ステップ１２０）。検査がＯＫであれば、アクセス制御リスト管理手段１３０が呼び出される。アクセス制御リスト管理手段はクライアント・インタフェース１９０を介してクライアントへアクセス制御リスト（後で説明する）を表示し、クライアントがアクセス制御リスト内の個々の項目を許可するか許可しないかを確かめる。アクセス制御リスト管理手段１３０は、クライアント・インタフェースを介してクライアントから命ぜられたとおりに、コード１４０を記憶し、許可フラグ１６０と共にアクセス制御リスト１５０を記憶する。

【００１２】アクセス制御リスト強制手段１８０によってアクセスを制御することのできるリソースには、ファイル・システム、特定のファイル、およびシステム呼び出しのような論理リソースと、ディスク・スペース、ディスク・アクセス、メイン・メモリの割り振り、および各種のシステム・コントローラやアダプタへのアクセスのような物理リソースがある。論理リソース（クライアント・システムの機能）へのアクセスについては、個々の項目が許可されるか許可されないかを、許可フラグ１６０が表示する。物理リソースへのアクセスについては、最大許容量または最大許容消費率を示すために許可フラグを使用できる。

【００１３】さらに、図２はクライアント・システムの異なった部分がどのように相互に動作するかを示す。複数ユーザ・クライアント・システムでは、各ユーザがそれぞれの許可フラグ群を持つことができる。さらに、アクセス制御リストの中には環境変数を入れることができる。実行中に環境変数を変えることによって、個々のユーザはアクセス特権をカスタマイズすることができる。アクセス制御リストと許可フラグはセキュリティ領域に記憶され、この領域の読み出しと更新はアクセス制御リスト強制手段１８０に許される特権である。

【００１４】アクセス制御リスト管理手段は、アクセス制御リストの許可条件を表示または変更するためにクライアントによって任意の時点で呼び出されてよい。コードは実行手段１７０によって実行される。リソースへのアクセスを許す前に、実行手段はアクセス制御リスト強制手段を呼び出して、アクセスの有効性をチェックする。これは、実行手段１７０が、アクセス制御リスト強制手段を呼び出す検査ルーチンへのトラップをコードの中に挿入するという方法によって達成される。アクセス制御リスト強制手段の動作は、後で詳細に説明する。

【００１５】システムでは、必要に応じてコードとそのアクセス制御リストを別個にまたは組み合わせてダウンロードすることができる。たとえば、コード作成システムの方針として、すべてのクライアントへアクセス制御リストを無料で提供するが、コード自体は有料にすることができる。

【００１６】図３はアクセス制御リストを示す。アクセス制御リストは２つの部分から構成される。すなわち、コードによって要求される物理リソースを含む物理リソース・テーブル２００と、コードによって要求される許可条件と論理リソースを含む論理リソース・テーブル２５０である。

【００１７】物理リソース・テーブル２００の中には、各リソースについての行があり、この行の中には物理リソース名２０５、リソース属性２１０、最大許容消費率２１５、および最大許容量２２０がある。リソース属性２１０は、物理装置またはリソース・ディスクに複数の属性があるときに使用される。たとえば、記憶装置についてスペースとＩ／Ｏ数があるような場合である。最大許容消費率２１５と最大許容量２２０は、リソースと属性の最大許容消費率と最大許容消費量である。

【００１８】論理リソース・テーブル２５０は、コードによって要求される外部ルーチン（論理リソースと呼ばれる）への各呼び出しについて１つの行を含む。各行には、論理リソース名２５５とパラメータ・リスト２６０がある。パラメータ・リスト２６０はパラメータ・エントリ２６５のリストを指し示す。各パラメータ・エントリ２６５は、有効なパラメータ範囲の集合を指定する。すなわち、その集合は組み合わせとして有効なパラメータ値の集合である。パラメータ・エントリ２６５の中には、次のパラメータ・エントリを指し示すｎｅｘｔＰＥフィールド２８０も含まれている。各パラメータのパラメータ範囲は２つのフィールドを含む。すなわち、パラメータ・タイプ２７０と、そのパラメータの有効範囲を指定するパラメータ値２７５である。ストリング・パラメータについては、パラメータ・タイプ２７０はＳＴＲであり、パラメータ値２７５はそのストリングについて有効な形式を指定する正規表現のリストである。整数パラメータについては、パラメータ・タイプ２７０はＩＮＴであり、パラメータ値２７５は整数範囲のリストである。パラメータ値２７５の中には環境変数の名前を入れることができ、その場合、実行時にその環境変数の値を置換することが想定されている。

【００１９】クライアントの中のアクセス制御リスト強制手段は、アクセス制御リストの中でコードについて指定された許可条件とリソースが提供され、許可条件／リソースの追加が許されないことを確実にする。

【００２０】アクセス制御リストの強制は静的であっても動的であってもよい。静的強制では、コードが実行される前に全面的に強制が行われ、コードが実行されている間の強制は必要とされない。動的強制では、コードが実行されている間に強制を行わねばならない。証明機関自体がアクセス制御リストを検査し、コードによる超過が生じないことが保証されれば、アクセス制御リストの強制機能はクライアント・システムで不必要となろう。

【００２１】図４は、アクセス制御リスト強制手段１８０によって使用されるデータ構造を示す。実行時物理リソース・テーブル３００には、各リソースについて１つの行がある。リソース名３００、リソース属性３１０、最大許容消費率３１５、および最大許容量３２０は、物理リソース・テーブル２００の対応するフィールドのコピーである。実際消費率３２５および実際使用量３３０の各フィールドは、それぞれ実行時におけるコードの実際の消費率と実際の消費量を追跡するために使用される。実行時論理リソース・テーブル３５０には、必要とされる各論理リソースについての行がある。実行時論理リソース・テーブルは論理リソース・テーブルのコピーであり、それに許可フラグが追加されている。許可フラグは、パラメータの有効なそれぞれの組み合わせについて、その組み合わせがアクセス制御リスト管理手段によって許可されているか許可されていないかを示す。論理リソース名３５５は、論理リソース・テーブル２５０の対応するフィールドのコピーであり、パラメータ・リスト３６０は実行時パラメータ・エントリ３６５のリストを指し示す。実行時パラメータ・エントリのパラメータ・タイプ３７０およびパラメータ値３７５は、論理リソース・テーブル内の対応するフィールドのコピーである。ｎｅｘｔＰＥフィールド３８０は次の実行時パラメータ・エントリ３６５を指し示し、許可フラグ３８５は、この実行時パラメータ・エントリ３６５が許可されているか許可されていないかを示すイエスまたはノーに設定されている。さらに、アクセス制御リスト強制手段はコード開始時点３９５を追跡する。

【００２２】図５は、論理リソースの許可条件がどのようにしてアクセス制御リスト強制手段によって強制されるかを示す。このパスは、コードが外部機能への呼び出しを行うとき常に呼び出される。ステップ４１０で、アクセス制御リスト強制手段１８０はパラメータの数、パラメータの値、および呼び出されつつある機能の名前の位置を決定する。その実際の方法は処理系によって異なる。たとえば、ジャバでは、これらはオペランド・スタック上に置かれている。ステップ４１５では、アクセス制御リスト強制手段は、実行時論理アクセス・テーブル３５０の中で、呼び出されつつある機能の行の位置を決定し、許可フラグがイエスに設定されている最初の実行時パラメータ・エントリ３６５の位置を決定する。機能名が見つからないか、そのような実行時パラメータ・エントリ３６５が存在しない場合、アクセス制御リスト強制手段はステップ４７０へ進み、呼び出しが許されないことを示して終了する。ステップ４２０では、アクセス

制御リスト強制手段は最初のパラメータの値の位置を決定して、それを現在のパラメータにする。ステップ４２５では、アクセス制御リスト強制手段は、現在のパラメータの値が実行時パラメータ・エントリ３６５によって許されているかどうかをチェックする。値が許されていれば、ステップ４３０で、アクセス制御リスト強制手段はもっとパラメータがあるかどうかをチェックする。さらにパラメータがあれば、ステップ４３５で、アクセス制御リスト強制手段は現在のパラメータを次のパラメータに設定しステップ４２５へ戻る。ステップ４３０で、それ以上のパラメータがないと、アクセス制御リスト強制手段はステップ４５０へ進み、呼び出しが許される旨を表示して終了する。

【００２３】ステップ４２５で、許容性のテストが失敗すると、アクセス制御リスト強制手段はステップ４４５へ進む。そこでは、アクセス制御リスト強制手段が実行時論理リソース・テーブル３５０をチェックして、許可フラグ３８５がイエスへ設定されている他の実行時パラメータ・エントリを探す。そのような実行時パラメータ・エントリがなければ、アクセス制御リスト強制手段はステップ４７０へ進み、呼び出しが許されない旨を表示して終了する。そのような実行時パラメータ・エントリ３６５があれば、アクセス制御リスト強制手段はステップ４２０へ進む。

【００２４】図６は、物理要件がどのようにしてアクセス制御リスト強制手段によって強制されるかを示す。アクセス制御リスト強制手段は、リソースを割り振る前にステップ５００で呼び出される。２つのパラメータ、すなわち要求されたリソース量（ＲＥＱＡＭＴ）と消費予想時間（ＣＯＭＰＴ）が提供される。ディスクＩ／Ｏについては、ＲＥＱＡＭＴはディスクＩ／Ｏの数であり、ＣＯＭＰＴはディスクＩ／Ｏが完了する予想時間である。ステップ５０５で、アクセス制御リスト強制手段は、このリソースについて実行時物理リソース・テーブル３００内の行を決定し、最大許容量３２０が指定されているかどうか、および実際使用量３３０にＲＥＱＡＭＴを加えたものが最大許容量３２０を超過するかどうかをチェックする。もし超過すれば、それはステップ５２７で消費が許されない旨を表示する。最大許容量３２０を超過しないとき、ステップ５１０で、アクセス制御リスト強制手段はこのリソースの予測消費率を、（実際使用量３３０　＋　ＲＥＱＡＭＴ）／（現時点　－　コード開始時点３９５　＋　ＣＯＭＰＴ）に従って計算する。ステップ５１５で、アクセス制御リスト強制手段は最大許容消費率３１５が指定されているかどうか、および計画された消費率が最大許容消費率３１５よりも大きいかどうかをチェックする。最大許容消費率３１５が指定されていないか、予測消費率がそれより大きくなければ、アクセス制御リスト強制手段は消費が許される旨を表示して終了する。予測消費率が大きければ、アクセス制御リスト強制手段は、ステップ５３０で、この動作を実行するのに必要な遅延を、（実際使用量３００　＋　ＲＥＱＡＭＴ）／最大許容消費率３１５　－　（現時点　－コード開始時点３９５　－　ＣＯＭＰＴ）に従って計算し、計算された遅延時間だけ消費が遅延しなければならない旨を表示するとともに、必要な遅延時間を戻す。

【００２５】リソースが消費された後、ステップ５５０で、アクセス制御リスト強制手段はリソース消費量（ＣＯＮＳＡＭＴ）を指定するパラメータを使用して呼び出される。呼び出されたアクセス制御リスト強制手段は、実際使用率３２５と実際使用量３３０を更新する。

【００２６】さらに、同じコードについて、受け取りシステムで、異なったユーザには異なったリソースと許容条件を割り振ることができる。それは、コードがインストールされる時点では、アクセス制御リスト強制手段により、異なったユーザへ与えられた特権に注目して、コードに許されたリソースと許容条件に特権を組み合わせることによって行うことができる。この場合、リソースと許容条件の集合は、各ユーザごとに別々に記憶されることが必要であろう。コードが実行されているときには、リソースと許容条件はユーザ・ベースで強制されることになろう。他方、コードの実行中にリソースと許容条件を決定する場合は、アクセス制御リスト強制手段により、異なったユーザへ与えられた特権に注目して、コードに許されたリソースと許容条件に特権を組み合わせることによって決定することができる。

【００２７】図７は、クライアント・システムで行われる完全性検査、実行、および強制の初期設定動作を示す疑似コードである。今までに説明した各種のテーブル、リスト、フラグその他のデータ構造はクライアント・システムのメモリ（たとえば、揮発性ランダム・アクセス・メモリ、ディスク、またはこれらの組み合わせ）の中でインスタンス化されることに注意されたい。前述したように、アクセス制御リスト強制手段とアクセス制御リスト管理手段は、プログラム・コードとして実施するのが望ましい。これらのプログラム・コードは、クライアント・システムのオペレーティング・システムへリンクされるか組み込まれる。図８はアクセス制御リスト管理手段の疑似コードを示す。図９はアクセス制御リスト強制手段の疑似コードを示す。

【００２８】実施例に基づいて本発明を説明してきたが、当業者による各種の変更と改善が可能であると思われる。したがって、本発明の実施例は単なる例であって、制限的に解釈されてはならないことを理解すべきである。本発明の範囲が請求範囲によって限定されることはいうまでもない。

【００２９】まとめとして、本発明の構成に関して以下の事項を開示する。

（１）プログラム・コードの配布方法であって、信頼さ

れる第三者の証明を受け取りシステムへ提供するステップを含み、上記プログラム・コードの検査された無害の動作に必要なリソースと許容条件を、上記信頼される第三者の証明の中にコンピュータで読み取り可能に記述したことを特徴とする、プログラム・コードの配布方法。

（２）上記（１）において、上記証明をプログラム・コードと共にカプセル化するステップを含む、プログラム・コードの配布方法。

（３）上記（１）において、上記受け取りシステムが上記証明を読み取るステップと、上記証明中に指定された許容条件を超過しないように上記受け取りシステムのリソースと許容条件を割り振るステップとを含む、プログラム・コードの配布方法。

（４）上記（２）において、上記証明に関連してユーザが選択したオプションに従って、上記受け取りシステムのリソースに対する上記プログラム・コードのアクセスと許容条件を否定または肯定するステップを含む、プログラム・コードの配布方法。

（５）上記（１）において、コンピュータで読み取り可能に記述した上記リソースと許容条件が暗号形式で上記受け取りシステムへ提供されることを特徴とする、プログラム・コードの配布方法。

（６）上記（１）において、暗号化された検査データが上記証明の中に含まれており、上記受け取りシステムが上記検査データを解読して上記リソースと許容条件の記述の完全性を検査することを特徴とする、プログラム・コードの配布方法。

（７）上記（３）において、必要なリソースの記述の中に、上記プログラム・コードによって使用される各リソースの量と、そのリソースの最大許容消費率の両方が記述されていることを特徴とする、プログラム・コードの配布方法。

（８）上記（３）において、必要な許容条件の記述の中に、上記プログラム・コードによってアクセスされる上記受け取りシステムの特定の機能が記述されていることを特徴とする、プログラム・コードの配布方法。

（９）上記（１）において、上記プログラム・コードの機能性が、上記第三者による検査に従って上記第三者の証明の中に記述されていることを特徴とする、プログラム・コードの配布方法。

（１０）上記（１）において、上記プログラム・コードが、サーバからプログラム・オブジェクトとしてダウンロードされるアプレットであることを特徴とする、プログラム・コードの配布方法。

（１１）上記（１）において、上記受け取りシステムの中で上記プログラム・コードの異なったユーザに対しては異なったリソースと許容条件を割り振ることを特徴とする、プログラム・コードの配布方法。

（１２）上記（１１）において、上記異なったユーザに許されるリソースと許容条件の組が、上記プログラム・

コードのインストール時に決定されることを特徴とする、プログラム・コードの配布方法。

（１３）上記（１１）において、上記異なったユーザに許されるリソースと許容条件の組が、上記プログラム・コードの実行時に決定されることを特徴とする、プログラム・コードの配布方法。

（１４）プログラム・コードの配布方法であって、信頼される第三者によるプログラム・コードの証明の中に、該プログラム・コードの検査された無害の動作に必要なリソースと許容条件をコンピュータで読み取り可能に記述したものを入れ、その証明を上記プログラム・コードと共にカプセル化して暗号形式で受け取りシステムへ提供するステップと、上記受け取りシステムが上記暗号形式の証明を読み取るステップと、上記読み取られた証明の完全性を上記受け取りシステムによって決定するステップと、上記完全性が検査された後に、上記証明の中で指定された許容条件を超過しないようにユーザが選択したオプションに従って上記受け取りシステムのリソースと許容条件を割り振るステップと、上記割り振りに従って上記プログラム・コードを実行するステップとを含み、上記必要なリソースの記述の中には、上記プログラム・コードによって使用される各リソースの量とその最大許容消費率との両方が記述され、上記許容条件の記述の中には、上記プログラム・コードによってアクセスされる上記受け取りシステムの特定の機能が記述されていることを特徴とする、プログラム・コードの配布方法。

（１５）コンピュータ・システムへプログラムとデータを移入する移入装置と、上記コンピュータ・システムの動作を制御するオペレーティング・システムと、コードの検査された無害の動作に必要なリソースをコンピュータで読み取り可能に記述したものを、上記データから抽出してそれを所与のプログラムに関連づけるアクセス・ロジックと、該アクセス・ロジックに含まれて、上記コンピュータで読み取り可能に記述したものの完全性を示す検査データを生成する完全性検査ロジックと、上記オペレーティング・システムに接続され、上記検査データに応答して多数のリソースの各々について上記受け取りシステムの中で消費量と消費率を追跡し割り振って、上記記述の中に指定された割り振りを超過しないようにする強制ロジックと、上記割り振りに従ってプログラム・コードを実行するプロセッサと、を具備するコンピュータ・システム。

（１６）上記（１５）において、ランダム・アクセス・メモリに記憶されたデータ構造が上記強制ロジックに接続されており、該データ構造の中には、消費された実際のリソースを追跡する第１のフィールドと、リソースの消費率を追跡する第２のフィールドと、上記記述から引き出されたリソース消費の限度を記憶する第３のフィールドと、上記記述から引き出されたリソース消費率の限度を記憶する第４のフィールドとが含まれていることを

特徴とする、コンピュータ・システム。

（１７）上記（１５）において、上記アクセス・ロジックの中に、プログラム・コードを含むパッケージから上記記述を非カプセル化する手段と該記述を解読する手段とが含まれていることを特徴とする、コンピュータ・システム。

（１８）上記（１５）において、上記強制ロジックの中に、上記記述に関連してユーザが選択したオプションに従ってコンピュータ・システムのリソースに対するコード・アクセスと許容条件を否定または肯定する手段が含まれていることを特徴とする、コンピュータ・システム。

【図面の簡単な説明】

【図１】本発明に従うコード配布・証明システムを示すブロック図である。

【図２】クライアント・システムの各部分が、実施例で説明される機能を実行するために、相互間でどのように動作するかを示す図である。

【図３】アクセス制御リスト（ＡＣＬ）を示す図である。

【図４】アクセス制御リスト（ＡＣＬ）強制手段によって使用されるデータ構造を示す図である。

【図５】論理リソースの許容条件がアクセス制御リスト（ＡＣＬ）強制手段によってどのように強制されるかを示す図である。

【図６】物理リソースの限度がアクセス制御リスト（ＡＣＬ）強制手段によってどのように強制されるかを示す図である。

【図７】クライアント・システムの完全性検査、実行、および強制の初期設定動作を疑似コードで示した図である。

【図８】アクセス制御リスト（ＡＣＬ）管理手段の疑似コードを示す図である。

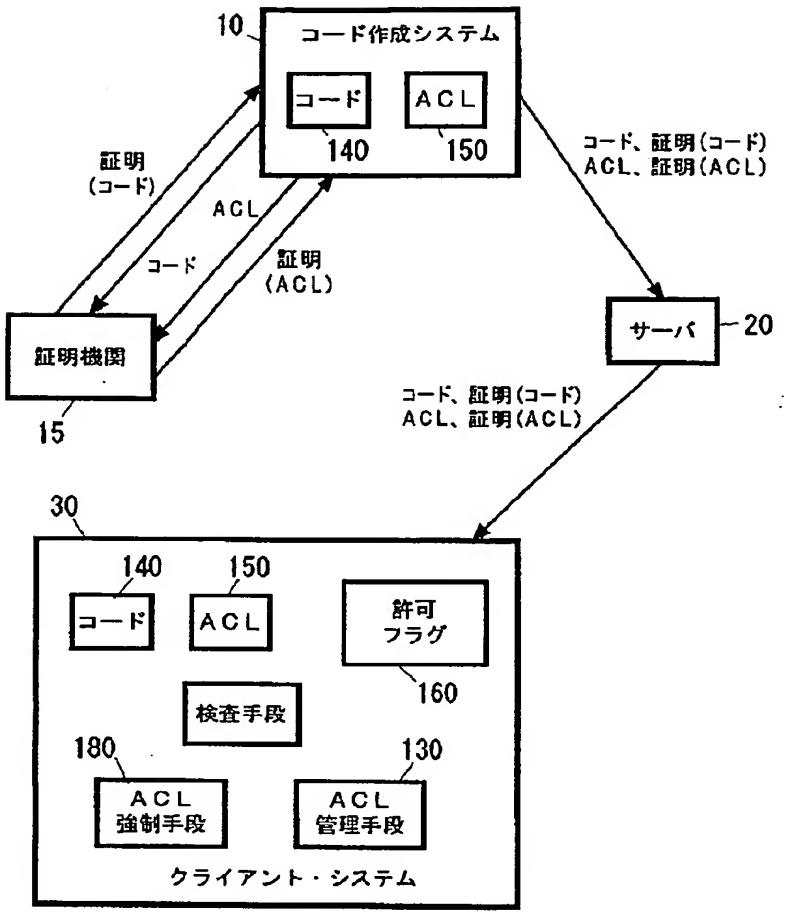
【図９】アクセス制御リスト（ＡＣＬ）強制手段の疑似コードを示す図である。

【符号の説明】

１０	コード作成システム
１５	証明機関
２０	サーバ
３０	クライアント・システム
１００	コード／アクセス制御リストおよび証明
１１０	検査手段
１２０	拒絶ステップ

１３０	アクセス制御リスト管理手段
１４０	コード
１５０	アクセス制御リスト
１６０	許可フラグ
１６２	ＣＰＵおよび関連制御ロジック
１７０	実行手段
１８０	アクセス制御リスト強制手段
１９０	クライアント・インタフェース
１９４	通信コントローラ／ネットワーク・インタフェース
１９６	ＣＤ－ＲＯＭ／フロッピ・ディスク・サブシステム
２００	物理リソース・テーブル
２０５	物理リソース名
２１０	リソース属性
２１５	最大許容消費率
２２０	最大許容量
２５０	論理リソース・テーブル
２５５	論理リソース名
２６０	パラメータ・リスト
２６５	パラメータ・エントリ
２７０	パラメータ・タイプ
２７５	パラメータ値
２８０	ｎｅｘｔＰＥ（次のパラメータ・エントリ）フィールド
３００	実行時物理リソース・テーブル
３０５	物理リソース名
３１０	リソース属性
３１５	最大許容消費率
３２０	最大許容量
３２５	実際消費率
３３０	実際使用量
３５０	実行時論理リソース・テーブル
３５５	論理リソース名
３６０	パラメータ・リスト
３６５	実行時パラメータ・エントリ
３７０	パラメータ・タイプ
３７５	パラメータ値
３８０	ｎｅｘｔＰＥ（次のパラメータ・エントリ）フィールド
３８５	許可フラグ
３９５	コード開始時点

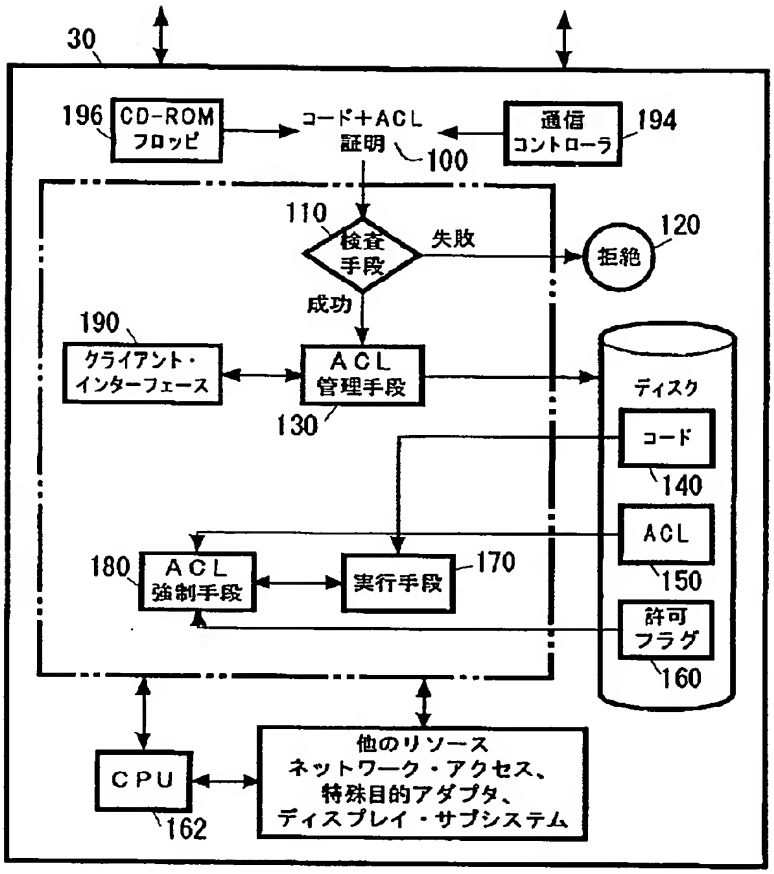
【図１】



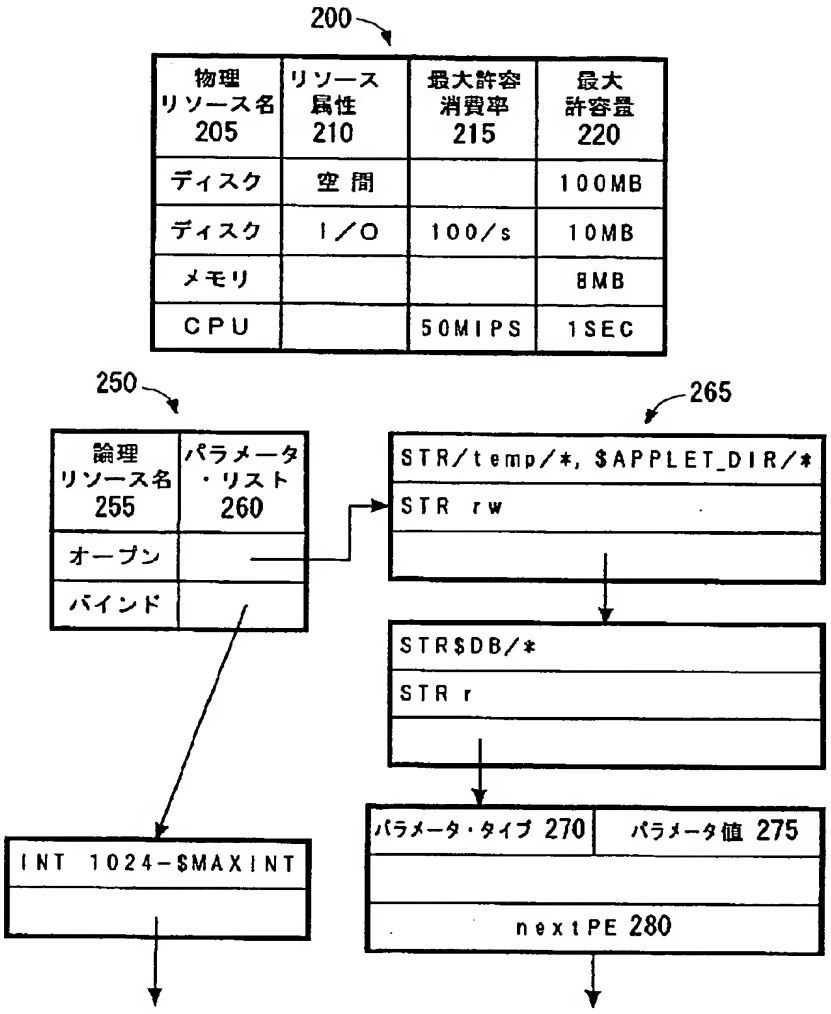
【図８】

```
if 新しく受取られたコードであれば then
        コードとＡＣＬをディスク上に記憶；
プログラムのＡＣＬを表示；
while まだクライアント入力があれば do
      クライアント入力を読取る；
      適切な許可フラグを変更する；
      変更された許可フラグをディスク上に記憶；
end
```

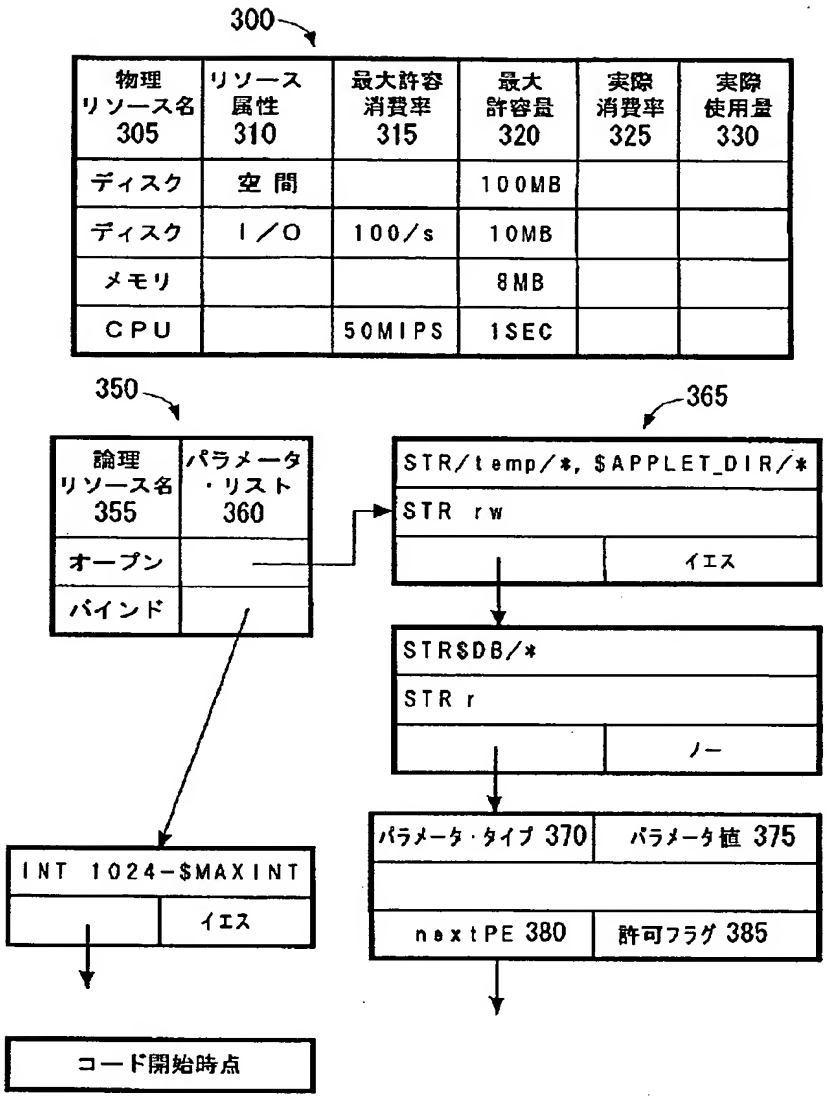
【図２】


30
196
CD-ROM
フロッピ
コード＋ACL
証明
100
通信
コントローラ
194
110
検査
手段
失敗
拒絶
120
成功
190
クライアント・
インターフェース
ACL
管理手段
130
ディスク
コード
140
ACL
150
許可
フラグ
160
180
ACL
強制手段
実行手段
170
CPU
162
他のリソース
ネットワーク・アクセス、
特殊目的アダプタ、
ディスプレイ・サブシステム

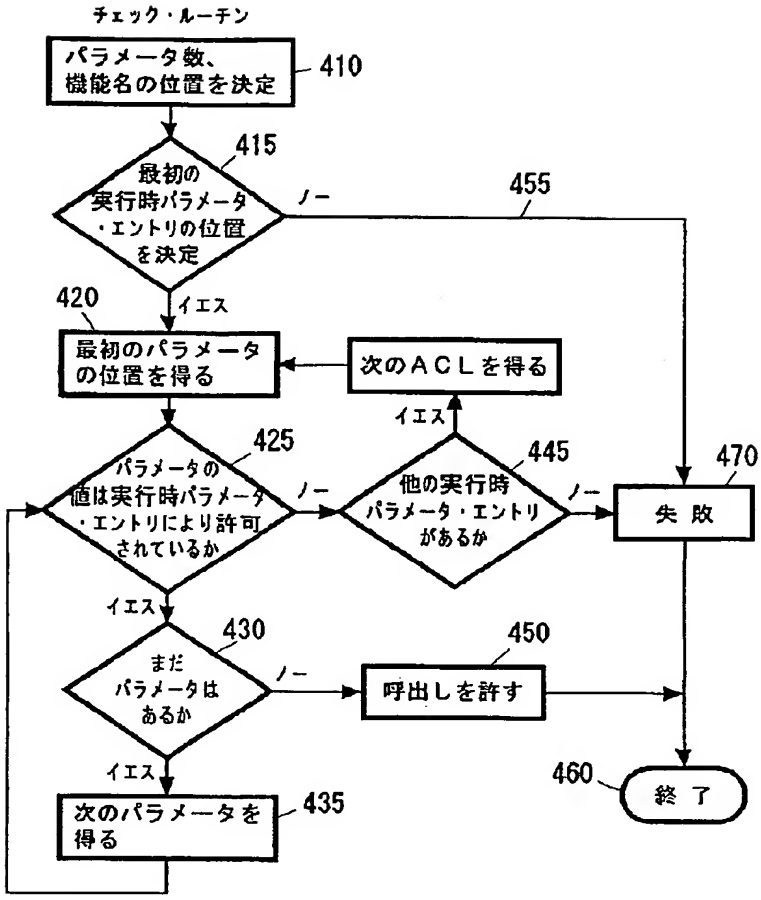
【図３】


200
物理リソース名 205
リソース属性 210
最大許容消費率 215
最大許容量 220
ディスク
空間
100MB
ディスク
I／O
100/s
10MB
メモリ
8MB
CPU
50MIPS
1SEC
250
論理リソース名 255
パラメータ・リスト 260
オープン
バインド
265
STR/temp/*, $APPLET_DIR/*
STR rw
STR$DB/*
STR r
INT 1024-$MAXINT
パラメータ・タイプ 270
パラメータ値 275
nextPE 280

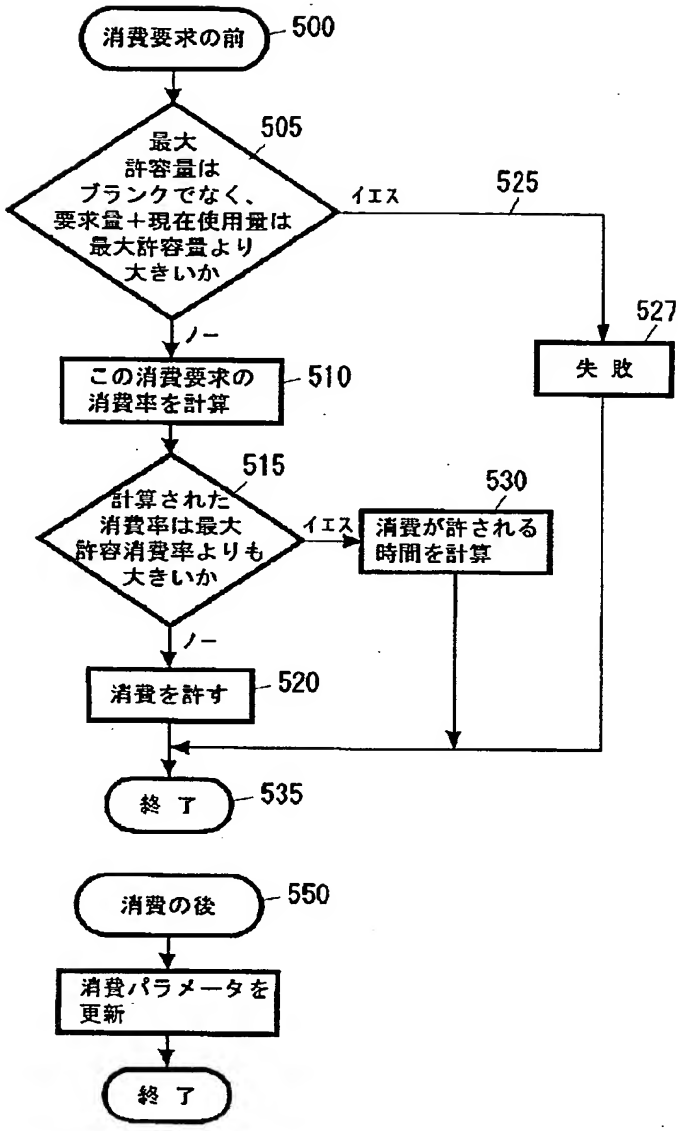
【図４】


300
物理リソース名 305
リソース属性 310
最大許容消費率 315
最大許容量 320
実際消費率 325
実際使用量 330
ディスク
空 間
100MB
ディスク
I／O
100/s
10MB
メモリ
8MB
CPU
50MIPS
1SEC
350
論理リソース名 355
パラメータ・リスト 360
オープン
バインド
365
STR/temp/*, $APPLET_DIR/*
STR rw
イエス
STR$DB/*
STR r
ノー
パラメータ・タイプ 370
パラメータ値 375
nextPE 380
許可フラグ 385
INT 1024-$MAXINT
イエス
コード開始時点

【図５】


チェック・ルーチン
パラメータ数、機能名の位置を決定
410
最初の実行時パラメータ・エントリの位置を決定
415
ノー
455
イエス
420
最初のパラメータの位置を得る
次のＡＣＬを得る
イエス
パラメータの値は実行時パラメータ・エントリにより許可されているか
425
ノー
他の実行時パラメータ・エントリがあるか
445
ノー
470
失敗
イエス
まだパラメータはあるか
430
ノー
呼出しを許す
450
イエス
次のパラメータを得る
435
460
終了

【図６】


消費要求の前
500
最大
許容量は
ブランクでなく、
要求量＋現在使用量は
最大許容量より
大きいか
505
イエス
525
ノー
この消費要求の
消費率を計算
510
失 敗
527
計算された
消費率は最大
許容消費率よりも
大きいか
515
イエス
530
消費が許される
時間を計算
ノー
消費を許す
520
終 了
535
消費の後
550
消費パラメータを
更新
終 了

【図７】

コードのインストール：

プログラム・コード、ＡＣＬおよび暗号ハッシュを受取る；
第三者の公用キーを使用して解読；
コードとＡＣＬからハッシュを計算；
ｉｆ 計算されたハッシュが解読されたハッシュと等しくなければ
ｔｈｅｎ コードを拒絶；
ｅｘｉｔ；
ＡＣＬ管理手段を呼出す；

コードの実行：

リソースがアクセスされるときアクセス検査へのトラップを
コード中に挿入；
変更されたコードを実行；

アクセス検査：

ＡＣＬ強制手段を呼出す；
ｉｆ アクセスが許されれば ｔｈｅｎ
継続；
ｅｌｓｅ ｉｆ アクセスが遅延されれば ｔｈｅｎ
プログラムを遅らせる；
ｅｌｓｅ ｉｆ アクセスが拒否されれば ｔｈｅｎ
プログラムを停止する；

【図９】

```
論理リソース・アクセスのチェック：
------------------------------
        この論理リソースの実行時パラメータ・エントリの位置を決定；
        ｗｈｉｌｅ まだ実行時パラメータ・エントリがあれば ｄｏ
            ｗｈｉｌｅ まだパラメータがあれば ｄｏ
                ｉｆ パラメータへのアクセスが許されなければ
                    ｇｏｔｏ ｇｅｔＮｅｘｔＲＰＥ；
            ｅｎｄ
            アクセス許可を戻す；

            ｇｅｔＮｅｘｔＲＰＥ： 次の実行時パラメータ
                                    ・エントリを得る；
        ｅｎｄ
        アクセス拒否を戻す；

物理リソース・アクセスのチェック：
------------------------------
        実行時物理リソース・テーブルの中でこの物理リソースの
        エントリの位置を決定；
        ｉｆ 最大許容量がブランクでなく、要求量＋使用量が
             最大許容量より大きければ ｔｈｅｎ
             アクセス拒否を戻す；
         予測消費率を計算；
        ｉｆ 予測消費率が最大許容消費率よりも大きければ ｔｈｅｎ
             必要な遅延を計算；
           アクセス遅延と必要な遅延を戻す；
        ｅｌｓｅ
           アクセス許可を戻す；
```

フロントページの続き

(72)発明者　ラジェヴィ・ラマスワミ
アメリカ合衆国10562、　ニューヨーク州
オッシニング　クロトン　アヴェニュー
52　アパートメント　６Ｃ

(72)発明者　ディンカール・シタラム
アメリカ合衆国10598、　ニューヨーク州
ヨークタウン　ハイツ　セス　レーン
525